به نام خدا

Housing Price Prediction

امیرحسین جهانخیر املشی , بابک رستمی , ماهر رسلان

SPRING 2020

مقدمه

پروژه Housing Price Prediction در رابطه با پیش بینی قیمت مسکن با استفاده از شبکه عصبی عمیق میباشد.

شبکه عصبی، دسته‌ای از الگوریتم‌هاست که برای شناسایی و تشخیص الگوها استفاده می‌شود. شبکه‌ی عصبی ساختاری شبیه به مغز انسان دارد و نامگذاری آن نیز به همین دلیل است. انواع داده‌ها مانند متن، تصویر، فیلم و ... قابل تبدیل به بردارهایی با مولفه‌های عددی هستند. الگوهایی که یک شبکه عصبی تشخیص می‌دهد، به شکل مولفه‌های عددی یک بردار چند بعدی هستند. بنابراین، شبکه عصبی، با استفاده از یک ابزار ادارکی ماشینی (machine perception)، داده‌های خام را دریافت می‌کند و سعی می‌کند الگوهای موجود در این داده‌ها را شناسایی کند.

با استفاده از الگوریتم های شبکه عصبی، می‌توان اقدام به دسته‌بندی (classification) یا خوشه‌بندی (clustering) داده‌ها کرد. شبکه عصبی را می‌توان مانند یک فیلتر دانست که داده‌های عبوری از خودش را دسته‌بندی یا خوشه‌بندی می‌کند. برای مثال، می‌توان مجموعه‌ای از داده‌های کلاس‌بندی‌شده (labeled) را به شبکه عصبی داد تا با این ورودی آموزش ببینید. سپس، می‌توان این انتظار را داشت که داده‌ای جدید به شکل خودکار کلاس‌بندی شوند.

دیتاست

به‌طورکلی تمامی روش‌ها و متدهای یادگیری ماشین بر روی داده‌ها اجرا می‌شوند و درواقع این داده‌ها هستند که نقطه شروع فرایند داده‌کاوی، علم داده‌ها و به‌طورکلی یادگیری ماشین می‌باشند.

دیتاست در این پروژه یک فایل csv شامل 300 عدد اطلاعات مربوط به قیمت خانه های فرضی میباشد که به صورت تصادفی توسط نرم افزار با زبان c++ تهیه شده اند.

متاسفانه در این پروژه نتوانستیم از دیتاست مربوط به خانه های شهر رشت استفاده کنیم چون قیمت ها به طرز غیر منطقی با یکدیگر اختلاف داشتند و باعث میشد شبکه عصبی به درستی یاد نگیرد.

	A	B	C	D	E	F	G
1	mahalle	masahat	parking	asansor	salsakht	otaghkhab	price
2	motahari	67	0	0	98	2	314.9
3	resalat	95	0	0	98	3	465.5
4	motahari	60	1	1	94	3	234
5	golestan	100	1	0	95	1	340
6	golshahi	88	1	0	95	3	325.6
7	golestan	60	1	0	96	2	246
8	golestan	89	1	0	98	1	427.2
9	resalat	76	1	0	96	2	311.6
10	motahari	112	1	0	97	1	481.6
11	motahari	68	1	1	94	2	251.6
12	golshahi	113	0	0	93	2	305.1
13	golshahi	127	0	0	95	1	406.4
14	resalat	115	0	0	93	1	299
15	motahari	129	1	1	93	2	438.6
16	golshahi	114	0	0	98	2	535.8
17	golshahi	123	0	1	98	2	639.6

بخشی از دیتاست استفاده شده در پروژه (قیمت به میلیون تومان میباشد)

این دیتاست شامل 12 محله فرضی زیر میباشد.

```
string a[] = { "golshahi","motahari","resalat","golestan" };
string b[] = { "emam ali","sahar","razi","boostan" };
 string c[] = { "sahand","pakdasht","ettehad","rezvan" };
```

که محله a از محله b و محله b از محله c دارای ارزش مکانی بالاتری میباشد.

برای تهیه این دیتاست یک قیمت پایه برای هر محله در نظر گرفته شد و با توجه به امکانات هر خانه این قیمت افزایش یافت.

مزیت این دیتاست نسبت به دیتاست خانه های شهر رشت در این است که قیمت ها به صورت منطقی به خانه ها تعلق گرفته است.

پیاده سازی شبکه عصبی

قدم اول

در ابتدا کتابخانه های موجود در پایتون برای ایجاد شبکه عصبی را به پروژه اضافه میکنیم.

```
import keras
import numpy as np
import pandas as pd
```

numpy مخفف numerical python است. Numpy کتابخانه ای است که می‌توان به کمک آن بر روی داده‌های عددی موجود در حافظه عملیات متنوعی انجام داد. آرایه های numpy شبیه به لیست های خود پایتون هستند ولی با این تفاوت که به گونه‌ای در حافظه ذخیره می‌شوند که می‌توان بر روی آن‌ها عملیات مختلفی را به صورت سریع‌تر انجام داد.

Pandas یکی از ابزار های محبوبی هست که در اکثر پروژه های مربوط به علم داده از ان استفاده میشود. در این پروژه نیز برای کار با فایل Csv و مرتب سازی داده ها از ان استفاده شده است

keras چهارچوبی است که با آن و تنها با چند خط کد می‌توانیم برای ساختن شبکه‌های عصبی استفاده کنیم. البته کراس همه این کارها را خودش به تنهایی انجام نمی‌دهد در حقیقت کراس یک (front-end) برای فریمورک های یادگیری عمیق تنسرفلو، CNTK است و آن‌ها پشت شبکه‌های عصبی را می‌سازند و آموزش می‌دهند و برای همین به آن یک چهارچوب سطح بالا می گوییم چون کراس پیچیدگی استفاده از این کتابخانه‌ها را تا حد خوبی حذف می‌کند. یک ویژگی خاص دیگر کراس این است که محدود به یک کتابخانه یادگیری عمیق نیست و همانطور که گفتیم می‌توانیم از تنسرفلو، CNTK و یا تیانو برای محاسبات پشت پرده آن استفاده کنیم.

لازم به ذکر است که فراگیری طرز استفاده از از این کتابخانه های ساخت شبکه عصبی عمیق و گرفتن خروجی مطلوب از ان ها کاری زمان بر است و در این مدت محدود ما توانستیم در حد نیاز طرز استفاده از این کتابخانه ها را فرا بگیریم.

قدم دوم

```
df=pd.read_csv('house.csv')
df.head()
```

با استفاده از کد بالا محتوای فایل csv را در متغیری به نام df ذخیره کردیم و با استفاده از تابع head پنج سطر اول متغیر df را مشاهده خواهیم کرد.

```
df.describe()
```

با استفاده از کد بالا نیز میتوانیم اطلاعات اماری از دیتاست خود را مشاهده کنیم.

دیتاست ما شامل نام محله برای هر خانه میباشد برای این که تمامی اطلاعات موجود در دیتاستمان به صورت عددی باشد نام محله ها را به ستون های دیتاست اضافه میکنیم و یک بودن عدد زیر ستون مربوط به نام محله ها نشان دهنده این است که ان خانه در ان محله قرار دارد.

برای انجام این کار کد های زیر را به ترتیب اجرا میکنیم.

```
dumm = pd.get_dummies(df.mahalle)
dumm.head(3)


df2=pd.concat([df,dumm],axis='columns')
df2.head()


df3=df2.drop('mahalle',axis='columns')
df3.head()
```

در نهایت df3 به صورت زیر در می اید:

```
df3=df2.drop('mahalle',axis='columns')
df3.head()
```

	masahat	parking	asansor	salsakht	otaghkhab	price	boostan	emam ali	ettehad	golestan	golshahi	motahari	pakdasht	razi	resalat	rezvan	sahand	sahar
0	67	0	0	98	2	314.9	0	0	0	0	0	1	0	0	0	0	0	0
1	95	0	0	98	3	465.5	0	0	0	0	0	0	0	0	1	0	0	0
2	60	1	1	94	3	234.0	0	0	0	0	0	1	0	0	0	0	0	0
3	100	1	0	95	1	340.0	0	0	0	1	0	0	0	0	0	0	0	0
4	88	1	0	95	3	325.6	0	0	0	0	1	0	0	0	0	0	0	0

مقدار نهایی دیتاست df3

برای این که تمامی اعداد موجود در دیتاست به صورت اعشاری باشند کد زیر را اجرا میکنیم

```
df3.isna().sum()
df3 = df3.astype(float)
df3.dtypes
```

قدم سوم

ما در این پروژه میخواهیم که قیمت خانه ها را پیشبینی کنیم بنابر این دیتاستمان را به دو بخش x و y تقسیم میکنیم که متغیر y فقط شامل ستون price خواهد بود و متغیر x همه ستون ها به غیر از ستون price را خواهد داشت.

برای این کار کد زیر را اجرا میکنیم.

```
x=df3.drop(columns=['price'])
y=df3[['price']]
```

برای ساخت شبکه عصبی توسط keras از کد زیر استفاده میکنیم

```
model=keras.models.Sequential()

model.add(keras.layers.Dense(32,activation='relu',input_shape=(17,)))
model.add(keras.layers.Dense(32,activation='relu'))
model.add(keras.layers.Dense(1))
```

```
model.compile(optimizer='adam',loss='mean_squared_error')
```

ماژول Sequential برای مقداردهی اولیه به ANN و ماژول Dense برای ساخت لایه‌های شبکه عصبی مصنوعی استفاده می‌شود.

در ادامه نیاز به مقداردهی اولیه به ANN با ساخت یک نمونه از Sequential است. تابع Sequential یک پشته خطی از لایه‌ها را مقداردهی اولیه می‌کند. این امر به کاربر امکان اضافه کردن لایه‌ها را در آینده و با استفاده از ماژول Dense می‌دهد.

از روش add برای افزودن لایه‌های مختلف به شبکه عصبی استفاده می‌شود. اولین پارامتر تعداد گره‌هایی است که کاربر قصد دارد به این لایه اضافه کند. هیچ قاعده ای برای محاسبه اینکه چه تعدادی گره باید اضافه کرد وجود ندارد. یک استراتژی متداول انتخاب تعداد گره‌ها برابر با میانگین تعداد گره‌های موجود در لایه ورودی و تعداد گره‌های لایه خروجی است.

برای مثال اگر پنج متغیر مستقل و یک خروجی وجود داشته باشد مجموع آن‌ها محاسبه شده و بر دو تقسیم می‌شود که برای مثال بیان شده برابر با سه است. همچنین می‌توان با روشی که (Parameter Tuning) نامیده می‌شود آزمایش کرد.

پارامتر بعدی یک (Activation Function) است. در اینجا از تابع یکسوساز که به صورت relu خلاصه شده به عنوان تابع فعال‌سازی استفاده می‌شود. اغلب از این تابع برای لایه پنهان در ANN استفاده می‌شود. پارامتر نهایی input_shape است که تعداد گره‌ها در لایه پنهان محسوب می‌شود. این پارامتر نشانگر تعداد متغیرهای مستقل است.

افزودن دومین لایه پنهان مشابه روش مورد استفاده برای افزودن اولین لایه پنهان است.

نیازی به تعیین پارامتر input_shape نیست زیرا در حال حاضر در اولین لایه پنهان مشخص شده است. در اولین لایه پنهان این متغیر مشخص شده تا به لایه این امکان داده شود که بداند انتظار چه تعدادی گره ورودی را داشته باشد. در دومین لایه پنهان، ANN می‌داند که باید انتظار چه تعدادی گره ورودی را داشته باشد بنابراین نیازی به تکرار این کار نیست.

در لایه اخر نیز فقط یک گره خروجی داریم.

در نهایت نیز شبکه عصبی را کامپایل میکنیم.

اولین پارامتر الگوریتمی است که مقرر شده از آن برای گرفتن مجموعه بهینه‌ای از وزن‌ها در شبکه عصبی استفاده شود. انواع گوناگونی از این پارامترها وجود دارند. یکی از الگوریتم‌های موثر برای این کار (Adam) است.

قدم چهارم (برازش)

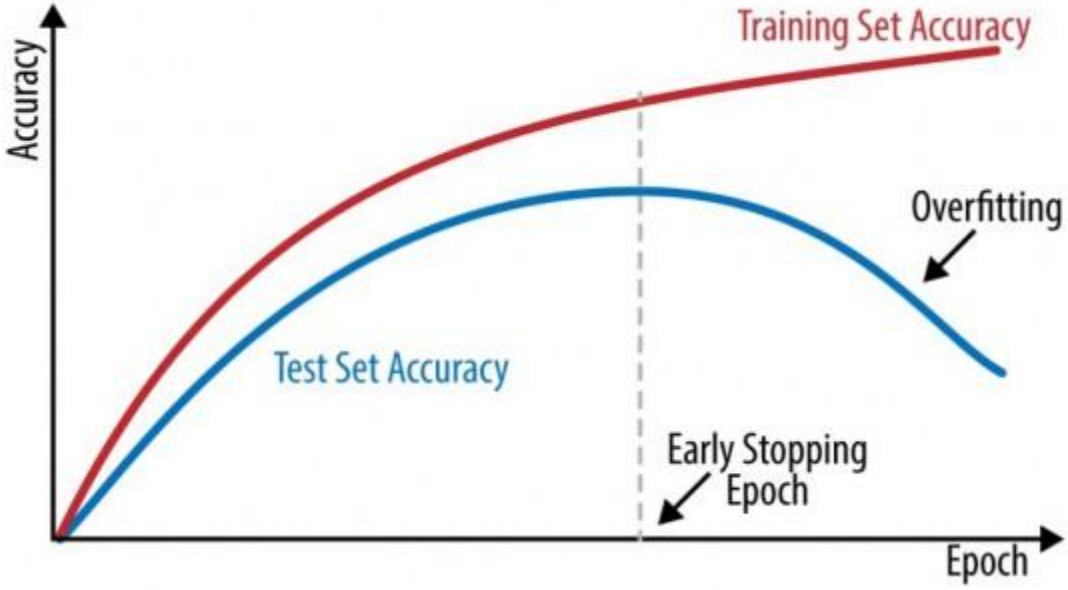
```
model.fit(x,y,epochs=30,callbacks=[keras.callbacks.EarlyStopping(patience=5)])
```

در کد بالا X متغیرهای مستقلی که برای آموزش شبکه عصبی مورد استفاده قرار می‌گیرند را نشان می‌دهد و y بیانگر ستونی است که پیش‌بینی می‌شود. دوره‌ها (Epochs) نشانگر تعداد دفعاتی هستند که مجموعه داده کامل به شبکه عصبی مصنوعی پاس داده خواهد شد.

همانطور که میدونید برای آموزش یک مدل داده (دیتا) رو به سه قسمت train/validation/test تقسیم میکنند.

EarlyStopping در واقع یک روش خودکار برای آموزش شبکه است زمانیکه شبکه عمق یا فضای لازم را جهت ارتقاء دقت نداشته باشه شبکه در حالت underfit قرار گرفته و اگر شبکه خیلی پیچیده تر از نیاز مسئله مورد نظر باشه شبکه شروع می

کنه به overtring یعنی در epoch های بعدی شبکه کم کم قدرت تعمیم خودشو از دست میده و شروع میکنه حفظ کردن بجای تعمیم .که دقت تو validation افت میکنه ما باید عملیات آموزش رو تا قبل از overfit شدن شبکه ادامه بدیم ، در واقع Early Stopping روشی خودکار جهت شناسایی این وضعیت هستش. شکل زیر به صورت شماتیک این نقطه را نشان میده.



قدم پنجم

```
test_data = np.array([67,0,0,99,2,0,1,0,0,0,0,0,0,0,0,0,0])
print(model.predict(test_data.reshape(1,17), batch_size=1))
```

در نهایت نیز میتوانیم با کد بالا یک خانه ساختگی را به شبکه عصبی بدهیم و قیمت پیشبینی شده توسط شبکه عصبی را مشاهده کنیم.

با تغییر در لایه های شبکه عصبی یا افزودن لایه ها و همچنین مقیاس کردن داده های ورودی میتوان به جواب های بهتر و دقیقتر رسید

پایان